15 Entwurf der Programmlogik

In diesem Kapitel werden die wesentlichen Entwurfentscheidungen für die Realisierung der drei Kernkomponenten *Selektion* (Kap. 15.1), *Abbildung* (Kap. 15.2) und *Auswertung* (Kap. 15.3) beschrieben.

15.1 Entwurf der Kernkomponente *Selektion*

15.1.1 Filtermodule

Das funktionale „Herz" der Kernkomponente *Selektion* besteht in der konfigurierbaren Generierung geeigneter graphischer Nutzerschnittstellen aus einem erweiterbaren und flexibel miteinander kombinierbaren Satz von Filtermodulen (vgl. Kap. 14.2.1), die jeweils zur Generierung einer intern weiterverarbeitbaren Beschreibung einer Teilbedingung aus interaktiven Nutzereingaben dienen. Filtermodule (im Folgenden auch kurz als Filter bezeichnet) werden dabei als Komponenten mit eigener graphischer Nutzerschnittstelle konzipiert, die autonom alle spezifischen Aspekte zwischen Anwender und Schnittstelle übernehmen, die zur Formulierung einer Teilbedingung erforderlich sind. Während individuelle Filtermodule aus Sicht des Anwenders jeweils die Nutzerschnittstelle zur Eingabe einer bestimmten, für ihn semantisch bedeutsamen Teilbedingung darstellen, fungiert jedes Filtermodul aus Sicht der Programmlogik als BlackBox zur Generierung von Anfragebestandteilen (vgl. Abb. 15.1).

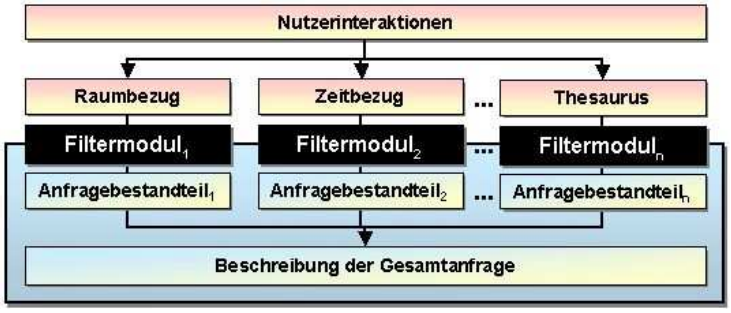

Abb. 15.1 - Filtermodule als Vermittler zwischen semantisch bedeutsamen Teilbedingungen für den Anwender und kombinierbaren Anfragebestandteilen für die Programmlogik.

Dazu stellt jedes Filtermodul unabhängig von seiner konkreten Ausprägung bestimmte Funktionalitäten in identischer Weise bereit, die den übrigen Programmteilen eine einheitliche Behandlung individueller Filtermodule ermöglichen:

- **Konfiguration**

Jedes Filtermodul kann in einheitlicher Weise von anderen Programmteilen durch Übergabe einer entsprechenden Beschreibung (Setup) konfiguriert werden, um für unterschiedliche Datenbankattribute und / oder Teilbedingungen verwendet werden zu können. Das Setup beinhaltet stets die Adaption des Filtermoduls an die konkreten Gegebenheiten der jeweils abzufragenden Datenbank und kann sich überdies sowohl auf die Gestaltung von Nutzerschnittstelle sowie Funktionalität des Filtermoduls beziehen. Das Setup wird in jedem Falle vom jeweiligen Filtermodul autonom analysiert und umgesetzt.

- **Vorliegen von Nutzereingaben**

Jedes Filtermodul kann in einheitlicher Weise von anderen Programmteilen aufgefordert werden, mitzuteilen, ob über seine graphische Schnittstelle Nutzereingaben vorliegen, die in eine Teilbedingung umzusetzen sind.

- **Korrektheit von Nutzereingaben**

Jedes Filtermodul kann in einheitlicher Weise von anderen Programmteilen aufgefordert werden, zu überprüfen und mitzuteilen, ob vorliegende Nutzereingaben gemäß der jeweils

gültigen syntaktischen und semantischen Vorgaben als korrekt anzusehen sind.

- **Generierung eines Anfragebestandteils**

Jedes Filtermodul kann in einheitlicher Weise von anderen Programmteilen aufgefordert werden, aus den vorliegende Nutzereingaben einen entsprechenden SQL-Anfragebestandteil zu generieren, der zur Konstruktion einer Gesamtanfrage verwendet werden kann, und diesen der Schnittstelle zur Verfügung zu stellen.

- **Rücksetzen**

Jedes Filtermodul kann in einheitlicher Weise von anderen Programmteilen aufgefordert werden, eventuell vorliegende Nutzereingaben zurückzusetzen.

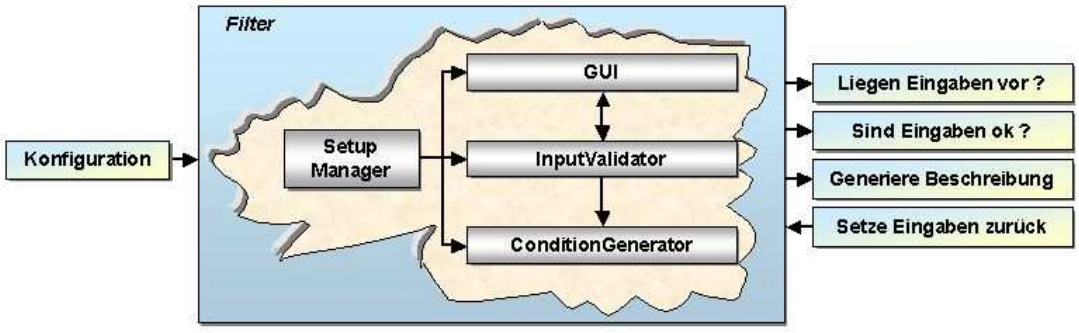

Abb. 15.2 - Die Architektur eines Filtermoduls und seine Schnittstellen zum Programm.

Jedes Filtermodul besteht zur Bereitstellung dieser Funktionalität aus vier zentralen Komponenten (vgl. Abb. 15.2), die eine weitgehende Entkopplung der wesentlichen Funktionseinheiten ermöglichen und somit seine flexible Realisierung und Adaptierbarkeit unterstützen. Diese Komponenten sind:

- **SetupManager**

Der SetupManager eines Filtermoduls stellt seine Adaptierbarkeit - und damit seine Wiederverwendbarkeit - sicher. Er analysiert dazu die dem Filtermodul übergebenen Initialisierungsparameter und steuert die Konfiguration der übrigen Bestandteile - graphische Nutzerschnittstelle, InputValidator und ConditionGenerator - jeweils entsprechend der aktuell geltenden Vorgaben.

- **Graphische Nutzerschnittstelle (GUI)**

Die graphische Nutzerschnittstelle eines Filtermoduls dient zur Kommunikation mit dem Anwender und erlaubt diesem die interaktive Eingabe der jeweiligen Teilbedingung. Jeweils vorliegende Nutzereingaben werden zur Überprüfung an den InputValidator weitergereicht.

- **InputValidator**

Der InputValidator eines Filtermoduls überprüft vorliegende Nutzereingaben auf syntaktische und semantische Korrektheit und reicht gegebenenfalls entsprechende Rückmeldungen über das GUI an den Anwender weiter.

- **ConditionGenerator**

Die Aufgabe des ConditionGenerator ist die Umwandlung der geprüften Nutzereingaben in eine entsprechend formulierten SQL-Bedingung.

Die konkrete Ausprägung dieser Komponenten bei einzelnen Filtermodulen ist für die mit diesen kommunizierenden anderen Programmteile unsichtbar. Auf diese Weise kann die Programmlogik der Schnittstelle vollständig von der individuellen Ausgestaltung einzelner Filtermodule sowie ihrer jeweiligen Semantik für den Anwender abstrahieren und bleibt von dieser unabhängig. Jedes vom Anwender für ein Selektionskriterium aktivierte Filtermodul generiert dabei einen SQL-Bestandteil, der nur eine der Teilbedingung entsprechende Un-

termenge des Datenraumes für das Abfrageergebnis zulässt. Da in jeder Anfrage alle ausgewählten Teilbedingungen erfüllt sein müssen, werden nur diejenigen Datensätze in ein Ergebnis aufgenommen, die sämtliche ausgewählten Teilbedingungen erfüllen.

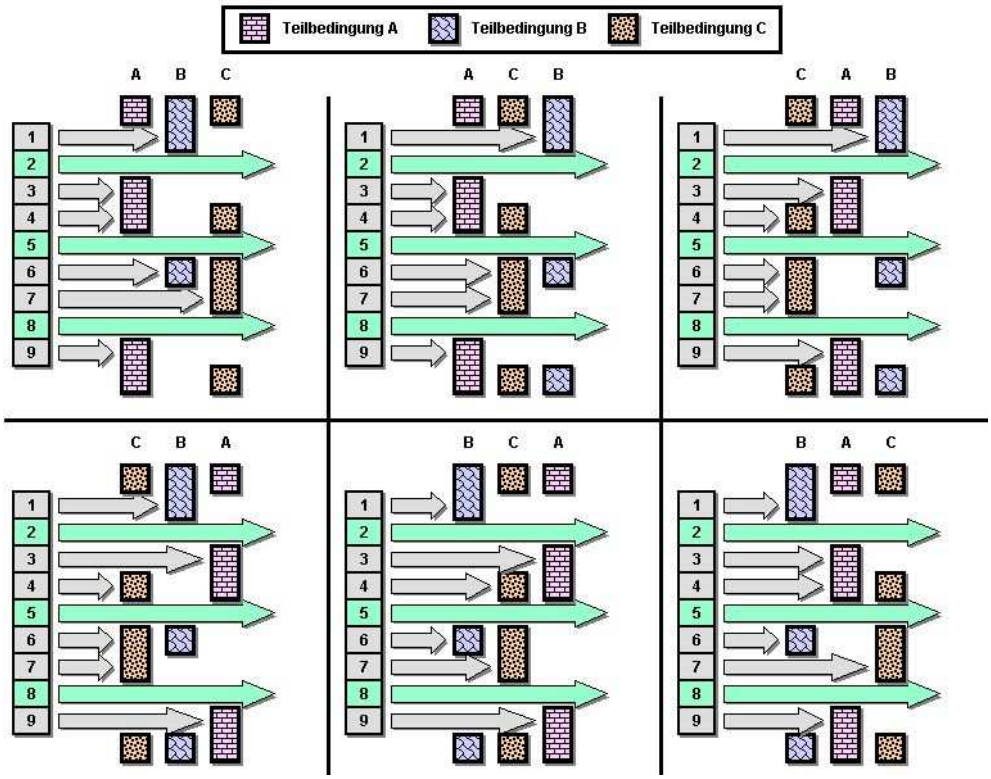

Abb. 15.3 - Teilbedingung fungieren wie Hindernisse, die jeweils nur eine spezifische Untermenge des Datenraumes passieren lassen. Jede Anordnung der Teilbedingungen A, B und C führt zum gleichen Ergebnis, im dargestellten Beispiel zur Auswahl der Datensätze 2, 5 und 8.

Die Reihenfolge, in der die jeweils ausgewählten Teilbedingungen auf einen Datenraum angewendet werden, ist dabei unerheblich. Abb. 15.3 verdeutlicht dies anhand aller sechs möglichen Kombinationen von drei Teilbedingungen A, B und C – unabhängig von der Reihenfolge, in der diese Teilbedingungen auf den Datenraum angewandt werden, wird jeweils die selbe Ergebnismenge selektiert. Die jeweils für eine Anfrage generierten SQL-Bestandteile der Teilbedingungen können daher ohne Berücksichtigung der Reihenfolge und ohne Hinzunahme weiterer Informationen mittels des $\wedge$-Operators (logisches UND) zu einer Gesamtanfrage verbunden werden (vgl. Kap. 15.1.3).

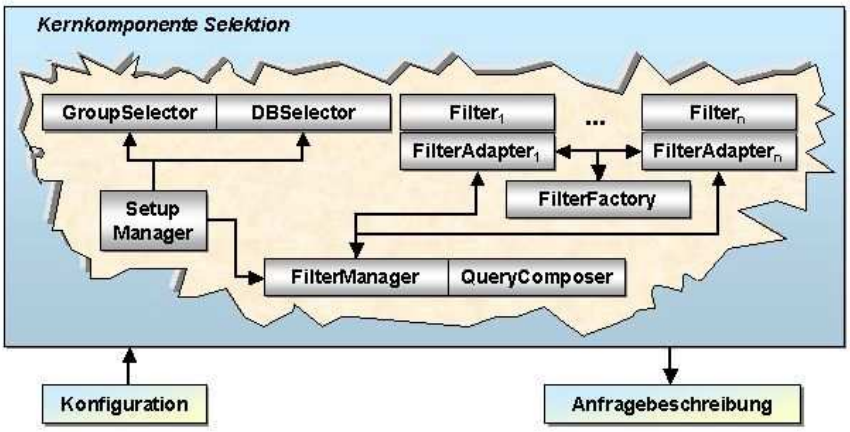

Abb. 15.4 - Architektur der Kernkomponente *Selektion*.

15.1.2 Architektur

Die generische Umsetzung der externen Konfigurationsvorgaben (vgl. Kap. 14.3.1) durch die Kernkomponente *Selektion* erfolgt durch das Zusammenwirken mehrerer Komponenten. Nachfolgendend werden die wesentlichen Bestandteile der gewählten Architektur (vgl. Abb. 15.4) überblicksartig beschrieben.

- **Verwaltung der Konfiguration**

Die aktuelle Konfiguration wird vom SetupManager der Kernkomponente analysiert, in entsprechende interne Repräsentationen transformiert und verwaltet. Sämtliche Informationen, die zur automatischen Identifikation und Konfiguration jeweils *eines* Filtermoduls erforderlich sind, werden dabei jeweils intern in transparent verarbeitbare sog. FilterSetups transformiert. In Kap. 14.3.1 wurde ausgeführt, dass die einzelnen zugänglichen Datenräume zur besseren Orientierung des Anwenders in Datenbankgruppen organisiert werden. Bevor die nutzerdefinierte Formulierung einer Anfrage stattfinden kann, sind daher nacheinander drei Schritte zu durchlaufen:

- die Auswahl einer Datenbankgruppe,
- die Auswahl eines Datenraumes aus dieser Datenbankgruppe sowie
- die Generierung einer Abfrageoberfläche für den ausgewählten Datenraum.

Zur Realisierung der verschiedenen Schritte bis hin zur Generierung einer Abfrageoberfläche aus den entsprechenden Filtermodulen kommuniziert der SetupManager nacheinander mit drei weiteren Komponenten – dem GroupSelector zur Auswahl einer Datenbankgruppe, dem DBSelector zur Auswahl eines Datenraumes sowie dem FilterManager zur Generierung einer entsprechenden Abfrageoberfläche (vgl. Abb. 15.5).

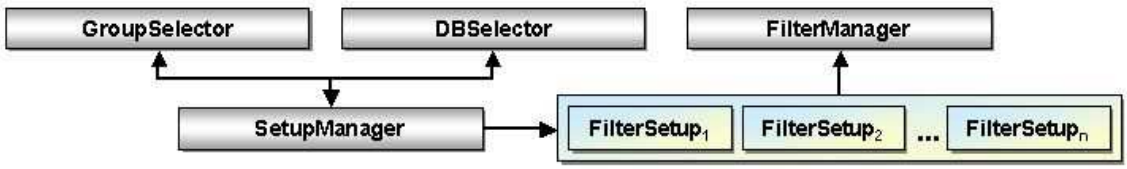

Abb. 15.5 - Konfigurierbare Auswahl von Datenbankgruppe und Datenbank.

- **Auswahl einer Datenbankgruppe**

Der SetupManager übergibt zunächst die Informationen über die entsprechend der Konfiguration selektierbaren Datenbankgruppen an die Komponente GroupSelector, die diese über eine entsprechende graphische Nutzerschnittstelle zugänglich macht[256].

- **Auswahl eines Datenraumes**

Hat sich der Anwender für eine Datenbankgruppe entschieden, übergibt der GroupSelector diese Information zurück an den SetupManager, der nun die dieser Gruppe zugeordneten Datenräume identifiziert und entsprechende Informationen über den DBSelector an den Anwender zur Auswahl weiterreicht[257].

- **Generierung der Abfrageoberfläche**

Sobald der Anwender einen der angebotenen Datenräume selektiert hat, reicht der DBSelector den entsprechenden Namen zurück an den SetupManager, der nun einen entsprechenden Satz von FilterSetups bereitstellt und zur Generierung der entsprechenden Abfrageoberfläche an den FilterManager (vgl. Abb. 15.6) übergibt. Der FilterManager dient zur Entkopplung der übrigen Programmlogik von allen Details der Verwaltung, Bereitstellung und Auswertung der Filtermodule für den jeweils ausgewählten Datenraum. Er wird über den vom SetupManager hierfür bereitgestellten Satz von FilterSetups konfiguriert und lie-

[256] Die graphische Oberfläche des GroupSelector ist Teil des Hauptfensters des Client (vgl. Kap. 19.2).
[257] Die graphische Oberfläche des DBSelector ist Teil des Hauptfensters des Client (vgl. Kap. 19.3).

fert schließlich die Beschreibung der vom Anwender anhand von Teilbedingungen formulierten Gesamtanfrage zurück. Zur Bereitstellung der Filtermodule und zur Entgegennahme der Anfragebestandteile generiert der FilterManager einen entsprechenden Satz von FilterAdaptern, die eine dynamische Initialisierung der erforderlichen GUI-Ressourcen erlauben; die Verknüpfung der Anfragebestandteile wird über den QueryComposer (vgl. Kap. 15.1.3) gesteuert.

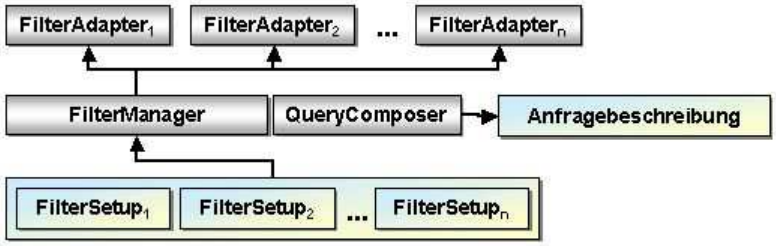

Abb. 15.6 - Von einem Satz von FilterSetups zur nutzerdefinierten Anfrage.

- **Dynamische Initialisierung der GUI-Ressourcen**

Je nach Leistungsfähigkeit des Anwenderrechners ist für die Initialisierung von graphischen Oberflächen eine gewisse Zeit erforderlich, die sich bei der gleichzeitigen Erzeugung vieler Filtermodule zu einer spürbaren und unerwünschten Verzögerung bei der Bereitstellung eines aktuellen Abfrage-Interface aufsummieren kann. Im gewählten Entwurf des Client (vgl. Kap. 19ff.) wird die graphische Oberfläche jedes Filtermoduls jeweils über ein eigenes Dialogfenster realisiert, das auf Anforderung des Anwenders geöffnet wird. Für deren effiziente Bereitstellung werden spezielle Komponenten, bezeichnet als FilterAdapter, verwendet, die es erlauben, die graphische Oberfläche eines Filtermoduls je nach Bedarf dynamisch zu initialisieren (vgl. Abb. 15.7).

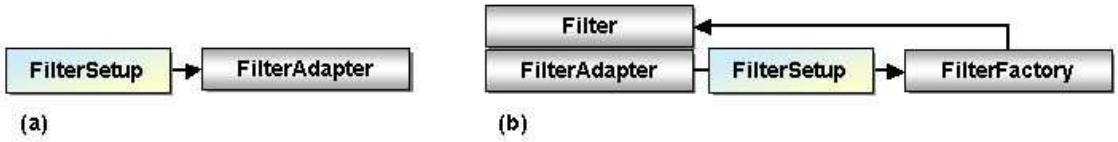

Abb. 15.7 - Die zwei Zustände eines FilterAdapters: (a) nach der Initialisierung über ein FilterSetup; (b) nach der Anforderung, das zugehörige Filtermodul zu öffnen.

FilterAdapter wurden als kleine generische Komponenten ohne eigene graphische Oberfläche konzipiert, die zur Laufzeit ohne spürbare Verzögerung erzeugt werden können. Jeder FilterAdapter kapselt dabei den Zugriff auf jeweils ein Filtermodul; er wird durch Übergabe eines FilterSetup initialisiert (vgl. Abb. 15.7a) und extrahiert aus diesem zunächst nur Namen und Benutzungshinweise, die so dem Anwender zur Orientierung zugänglich gemacht werden können. Das eigentliche Filtermodul wird hingegen erst - und nur dann - generiert, wenn es durch entsprechende Nutzerinteraktionen angefordert wird (vgl. Abb. 15.7b). Auf diese Weise kann der Aufwand zur Erzeugung der graphischen Oberflächen jeweils auf diejenigen Filtermodule beschränkt werden, die ein Anwender für die Formulierung seiner Anfrage auch tatsächlich benötigt. Für den internen Zugriff auf die gekapselte Funktionalität des Filtermoduls stellen FilterAdapter entsprechende identisch gestaltete Schnittstellen bereit, so dass sie programmseitig in gleicher Weise verwendet werden können wie Filtermodule selbst.

- **Transparente Generierung benötigter Filtermodule**

Zu Kapselung aller Details der Generierung individueller Filtermodule findet deren Erzeugung und Konfiguration in einer eigenen Komponente, der FilterFactory, statt, an die ein FilterAdapter bei Bedarf sein FilterSetup übergibt (vgl. Abb. 15.8). Die FilterFactory analysiert das Setup, identifiziert so das angeforderte Filtermodul und reicht das Setup an dieses

weiter. Das Filtermodul konfiguriert sich entsprechend und wird anschließend an die aufrufende Komponente übergeben, die ihrerseits vollständig von der konkreten Ausprägung des Filtermoduls abstrahieren kann. Die FilterFactory erlaubt es ferner, Filtermodule auch bereits vor ihrem eigentlichen Einsatz zu erzeugen[258] sowie einmal generierte Filtermodule vorzuhalten, um sie auf Anforderung schneller bereitstellen zu können.

Abb. 15.8 - Transparente Bereitstellung konkret erforderlicher Filtermodule aus einem FilterSetup.

15.1.3 Von der Konfiguration zur Generierung der Gesamtanfrage

Abschließend sollen kurz die verschiedenen Phasen überblicksartig dargestellt werden, die von der dynamischen Konfiguration des FilterManager bis zur Generierung einer Gesamtanfrage aus den vorliegenden Nutzereingaben durchlaufen werden:

- **Konfiguration**

Bei erfolgter Auswahl eines konkreten Datenraumes durch den Anwender bestimmt der SetupManager anhand der Konfigurationsvorgaben für die Kernkomponente *Selektion* (vgl. Kap. 14.3.1) einen Satz von FilterSetups und reicht diesen an den FilterManager weiter. Dieser generiert einen entsprechenden Satz von FilterAdaptern, denen jeweils ein FilterSetup übergeben wird. Namen sowie Informationen über die Bedeutung der einzelnen FilterModule werden durch Kommunikation mit den FilterAdaptern bereitgestellt.

- **Aufruf eines individuellen Filters**

Wählt der Anwender ein konkretes FilterModul aus, leitet der FilterManager diese Aufforderung an den entsprechenden FilterAdapter weiter. Dieser überprüft, ob das von ihm verwaltete FilterModul bereits erzeugt wurde; ist dies der Fall, stellt er dessen graphische Nutzerschnittstelle zur Verfügung. Wird das FilterModul hingegen zum ersten Mal angefordert, übergibt der FilterAdapter zunächst das von ihm verwaltete FilterSetup an die FilterFactory, um ein entsprechendes Filtermodul zurückzuerhalten.

- **Generierung der Gesamtanfrage**

Der FilterManager kann von der Programmlogik der Schnittstelle zu einem beliebigen Zeitpunkt aufgefordert werden, die aktuell vorliegenden Teilbedingungen einzusammeln und aus ihnen eine Beschreibung der Gesamtanfrage zu generieren. Er geht dazu alle FilterAdapter der Reihe nach durch und stellt zunächst fest, ob über das entsprechende FilterModul eine Teilbedingung vorliegt; ist dies der Fall, wird die entsprechend formulierte Anfragebedingung angefordert[259]. Die Konstruktion einer Gesamtanfrage aus den vorliegenden SQL-Bestandteilen der Teilbedingungen erfolgt anschließend durch ihre Verknüpfung anhand logischer Operatoren. Typischerweise entspricht hier eine Verknüpfung aller vorliegenden Teilbedingungen TB_i mit dem $\wedge$-Operator (logisches UND) in der Form

$$TB_1 \wedge TB_2 \wedge TB_3 \wedge ... \wedge TB_n$$

der Erwartungshaltung der Anwender. Aufgrund des gewählten Konzeptes ist es grundsätzlich ebenso möglich, die einzelnen Teilbedingungen, die dem FilterManager ja getrennt vorliegen, flexibel unter Verwendung weiterer logischer Operatoren wie dem $\vee$-Operator

[258] Dies wird für aufwendiger zu initialisierende Filtermodule eingesetzt. Ein Beispiel hierfür ist der SpatialFilter (vgl. Kap. 20.3) zur Auswahl von Raumbezügen über den interaktiven digitalen Atlas IDA. Hier muss beim Öffnen des Filtermoduls bereits eine der Konfiguration entsprechende Karte in IDA vorliegen.

[259] Da die graphischen Oberflächen der Filtermodule jeweils als Dialogfenster realisiert wurden, findet die Validierung der Nutzereingaben bereits bei der Aktivierung der Teilbedingung durch Schließen des entsprechenden Fensters statt (vgl. Kap. 20.1.3).

und dem $\neg$-Operator (logisches ODER bzw. Negation) sowie von Klammern zur Gruppierung zu verknüpfen, um die Formulierung komplexerer Anfragen bspw. der Form

$$(TB_1 \wedge \neg TB_2) \vee (TB_2 \wedge \neg TB_3) \ldots \wedge TB_n$$

zu unterstützen. Um entsprechende Funktionalität bei Bedarf einbetten zu können, wurde der QueryComposer integriert, der die Schnittstelle des FilterManager zur jeweils zu verwendenden Art der Verknüpfung darstellt. Der aktuell verwendete QueryComposer realisiert eine automatische UND-Verknüpfung aller für eine Anfrage vorliegenden Teilbedingungen, kann jedoch bei Bedarf durch einen gezielten Eingriff in den Programmcode unaufwendig gegen eine entsprechende Komponente für die interaktive Eingabe komplexer Verknüpfungen ausgewechselt werden[260].

15.2 Entwurf der Kernkomponente *Abbildung*

Entsprechend der gewählten Drei-Schichten-Architektur ist die Kernkomponente *Abbildung* im Server-Layer der Schnittstelle angesiedelt. Ihre Aufgabe ist die transparente Bereitstellung der vom Anwender über die Kernkomponente *Selektion* definierten Untermenge eines Datenraumes. Dies geschieht durch Umsetzung der von dieser generierten Anfragebeschreibung in eine entsprechende Datenextraktion und ihre Transformation in eine nachfolgend von der Kernkomponente *Auswertung* weiter zu verarbeitende Beschreibung dieser Daten.

15.2.1 Konfiguration

Um den Zugriff auf bisher nicht adressierte relationale Datenbankmanagementsysteme (RDBMS) eröffnen zu können, ohne in den Quellcode eingreifen zu müssen, kann der Server der Schnittstelle - und mit ihm die Kernkomponente *Abbildung* - ebenfalls über eine externe Konfigurationsdatei adaptiert werden. Die Konfigurationsinformationen umfassen bspw.:

Abbildung auf RDBMS	‣ Für jeden in die Schnittstelle eingebundenen Datenraum kann festgelegt werden, auf welches RDBMS entsprechende Anfragen abzubilden sind.
Abbildung auf Datenbanken	‣ Für individuelle Datenräume können bei Bedarf mehrere Datenbanken mit unterschiedlichem Detaillierungsgrad eingebunden werden, auf die eine Anfrage dann dynamisch abgebildet wird (vgl. Kap. 16.2.1, Tabellenkaskaden).
RDBMS-Server	‣ Für jedes zu adressierende RDBMS können die erforderlichen Zugriffsinformationen konfiguriert werden; hierunter fallen bspw. Internetadresse, Portnummer und Name des jeweiligen RDBMS-Servers.
JDBC-Treiber	‣ Für jedes zu adressierende RDBMS kann der zu verwendende JDBC-Treiber festgelegt werden.
Anzahl der Verbindungen	‣ Für jedes zu adressierende RDBMS kann festgelegt werden, wie viele Verbindungen zu dessen Server gleichzeitig zu öffnen sind (vgl. Kap. 15.2.2).
Ergebnismengen	‣ Da einzelne Datenbanken im gegebenen Kontext große Daten-

[260] Für die flexible nutzerdefinierte Kombination von Teilbedingungen wurde im Rahmen dieser Arbeit auch ein entsprechendes graphisches Interface entwickelt, das die schnelle Definition entsprechender Verknüpfungen für die jeweils aktuellen Teilbedingungen ermöglicht. Dieses Feature wurde in Betriebsphase I bereitgestellt, in der gegenwärtigen Version der Schnittstelle jedoch im Interesse einer möglichst intuitiven Anfrageerstellung allerdings wieder deaktiviert, da es von den Anwendern kaum genutzt wurde.

mengen enthalten können, kann für jede einzelne Datenbank die maximale Anzahl von pro Anfrage bereitzustellenden Datensätzen[261] festgelegt werden. Der Server limitiert dann entsprechend das jeweils bereitzustellende Datenvolumen für jedes Ergebnis.

15.2.2 Architektur

Nachfolgend werden die wesentlichen Bestandteile der Kernkomponente *Abbildung* überblicksartig vorgestellt (vgl. Abb. 15.9).

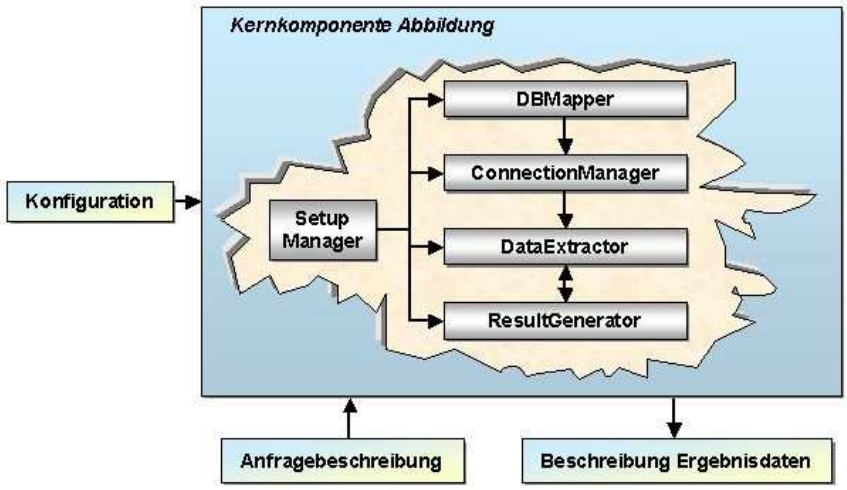

Abb. 15.9 - Architektur der Kernkomponente *Abbildung*.

- **Umsetzung der Konfiguration – SetupManager**

Der SetupManager der Kernkomponente Abbildung dient zur Analyse, Verwaltung und Umsetzung der externen Konfigurationsinformationen und zur entsprechenden Adaption ihrer weiteren Komponenten ConnectionManager, DBMapper, DataExtractor und ResultGenerator.

- **Verwaltung der Datenbankverbindungen – ConnectionManager**

Entsprechend der vorliegenden Konfigurationsinformationen werden beim Start des Servers für jedes einzubindende relationale Datenbankmanagementsystem (RDBMS) die entsprechenden JDBC-Treiber geladen und die jeweils vorgegebene Zahl von Verbindungen zu den einzelnen RDBMS-Servern aufgebaut. Diese Aufgabe übernimmt der ConnectionManager, der die aufgebauten Verbindungen vorhält und bei Bedarf für Anfragen bereitstellt.

- **Identifikation des relevanten RDBMS – DBMapper**

Der DBMapper dient zur Identifikation desjenigen RDBMS, das für eine eingehende Anfrage zu kontaktieren ist. Dazu wird eine ankommende Anfragebeschreibung ausgewertet, eine geeignete Verbindung identifiziert und beim ConnectionManager angefordert. Die Verbindung wird gemeinsam mit der Anfragebeschreibung an den DataExtractor weitergereicht, der für den entsprechenden Zugriff auf das RDBMS verantwortlich ist.

- **Zugriff auf die RDBMS – DataExtractor**

Der DataExtractor dient zur Kommunikation mit dem jeweils für eine Anfrage anzusprechenden RDBMS. Unter Nutzung der ihm hierzu übergebenen Datenbankverbindung sendet er die Anfrage an den entsprechenden RDBMS-Server und erhält von diesem die daraufhin selektierten Daten zurück. Die Daten werden anschließend zur Transformation an

[261] Die Schnittstelle unterstützt einen schrittweisen Zugriff auf größere Ergebnismengen (vgl. Kap. 21.2.3).

den ResultGenerator weitergeleitet. Der DataExtractor wurde so entworfen, dass mehrere Datenbankanfragen nebenläufig verarbeitet und an die entsprechenden RDBMS-Server weitergeleitet werden können. Da so zeitnah abzuarbeitende Anfragen an unterschiedliche RDBMS auf die jeweiligen RDBMS-Server verteilt werden können, kann die dort stattfindende Datenextraktion quasi parallel erfolgen, so dass in diesen Fällen eine performantere Abarbeitung von Nutzeranfragen möglich ist. Ein Performance-Gewinn entsteht auch bei der zeitnahen Weiterleitung mehrerer Anfragen an ein RDBMS, sofern dieses bspw. auf einer Mehrprozessor-Hardware aufsetzt und Parallelabarbeitung unterstützt.

- **Ergebnisaufbereitung und multiple Anfragen – ResultGenerator**

Die Umwandlung jeweils selektierter Daten in die entsprechende interne Ergebnisrepräsentation (vgl. Kap. 15.2.3) übernimmt der ResultGenerator. Diese Transformation erfolgt bereits beim Server, da auf diese Weise mehrere Vorteile zu erzielen sind. Zunächst muss so die hierfür erforderliche Logik nicht im Client implementiert werden, so dass eine weniger komplexe und umfangreiche Client-Software erforderlich ist. Die verwendete Ergebnisrepräsentation führt zudem in vielen Fällen zu einer Datenreduktion, so dass das zum Client zu übertragende Datenvolumen verringert wird.

Für bestimmte Anfragen kann es je nach vorliegender Datenbankmodellierung erforderlich sein, temporäre Datenbanktabellen zu generieren, wenn die Selektionskriterien allein anhand von SQL-Statements ausgedrückt werden sollen. Um dies zu umgehen, findet in solchen Fällen eine entsprechende Ergebnisaufbereitung durch den ResultGenerator statt; dabei werden gegebenenfalls autonom zusätzliche Anfragen ausgeführt, um ein hinreichend umfangreiches Anfrageergebnis bereitstellen zu können[262].

15.2.3 Adaptive interne Ergebnisrepräsentation

Für einen flexiblen Zugriff der Kernkomponente *Auswertung* auf beliebigen Ergebnisdaten wurde eine adaptive interne Repräsentation entwickelt, die jeweils die Daten eines Anfrageergebnisses organisiert vorhält und den Zugriff auf diese ermöglicht. Die interne Ergebnisrepräsentation stellt unabhängig von der Ausprägung des jeweiligen Anfrageergebnisses generisch einen Satz an Basisfunktionalität über diesem bereit, der von den jeweils aktiven Auswertungsmodulen in Anspruch genommen werden kann. Dazu zählen:

- *Sortieren* nach bis zu drei Ergebnisattributen zugleich, wobei für jedes Attribut jeweils zwischen Auf- und Abwärtssortierung gewählt werden kann. Da in der internen Ergebnisrepräsentation Daten in einer 1:n-Repräsentation verwaltet werden können (vgl. u.), wird für jeden einzelnen Datensatz zudem ein getrenntes Sortieren von Attributen, für die jeweils multiple Werteausprägungen vorliegen können, unterstützt;
- *Selektion und Deselektion* von Untermengen der repräsentierten Daten anhand direkter Angabe von Indizes sowie dynamisch anhand der Angabe von Selektionskriterien über die Werte mehrerer Attribute zugleich;
- *Aktivieren und Deaktivieren* von Untermengen der repräsentierten Daten[263];
- *Flexibler Export* von definierbaren Untermengen der repräsentierten Daten, um sie auf einen Datenträger des Anwenders zu speichern.

[262] Eine entsprechende Situation ist bspw. bei den vorliegenden Strukturen der angebundenen Zeitreihendatenbanken dann gegeben, wenn eine Anfrage die Identifikation solcher Stationen verlangt, die einen nutzerdefinierten Satz von Variablen - und nicht nur mindestens eine dieser Variablen - erheben.

[263] Diese Funktionalität kann zur dynamischen Reduktion der Ergebniskomplexität verwendet werden, bspw. durch nutzerdefiniertes Ausblenden von Teilen des Ergebnisses aus der Darstellung (vgl. Kap. 21.2.3).

Während das relationale Datenmodell durch Normalisierung eine weitgehend redundanzfreie Speicherung von Daten anhand mehrerer miteinander verknüpfter Tabellen ermöglicht, können aus Abfragen - oder bereits zuvor durch die Abbildung in eine Tabelle - durch Kreuzprodukte über die angesprochenen Datenbanktabellen entsprechend größere Ergebnistabellen resultieren, in denen Daten für einige der Attribute teilweise redundant vorliegen.

(a)

Stat_ID	Stat_Name	Stat_Type	Stat_Lon	Stat_Lat	Stat_Alt
4711	StationX	meteorology	22.22	33.33	999
…	…	…	…	…	…

Stat_ID	Var_ID	Begin_Date	End_Date
4711	123	1.1.1900	4.4.2003
4711	246	5.8.1922	6.6.2002
4711	777	1.1.1900	9.1.1988
…	…	…	…

Var_ID	Var_Name	Var_Unit
123	temperature	°C
246	precipitation	mm
777	wind speed	m/s
876	air pressure	hPa
…	…	…

⇩⇩⇩⇩⇩⇩⇩⇩⇩

(b)

Stat_ ID	Stat_ Name	Stat_Type	Stat_ Lon	Stat_ Lat	Stat_ Alt	Variable	Begin_ Date	End_ Date
4711	StationX	meteorology	22.22	33.33	999	temperature	1.1.1900	4.4.2003
4711	StationX	meteorology	22.22	33.33	999	precipitation	5.8.1922	6.6.2002
4711	StationX	meteorology	22.22	33.33	999	wind speed	1.1.1900	9.1.1988
…	…	…	…	…	…	…	…	…

⇩⇩⇩⇩⇩⇩⇩⇩⇩

(c)

Stat_ ID	Stat_ Name	Stat_Type	Stat_ Lon	Stat_ Lat	Stat_ Alt	Variable	Begin_ Date	End_ Date
4711	StationX	meteorology	22.22	33.33	999	temperature	1.1.1900	4.4.2003
						precipitation	5.8.1922	6.6.2002
						wind speed	1.1.1900	9.1.1988
						…	…	…

Abb. 15.10 - Unterschiedlich redundante Repräsentationen von Zeitreihenmetadaten: (a) redundanzfreie Repräsentation in der Datenbank (vereinfachte Darstellung); (b) teilweise redundantes Abfrageergebnis; (c) weitgehend redundanzfreie interne Ergebnisrepräsentation (schematische Darstellung).

So liegen bei Zeitreihenmetadaten 1:n-Relationen zwischen den Daten zur Charakterisierung einer Station (bspw. Stationsname und Georeferenzierung) und den Informationen zur Beschreibung der an dieser Station erhobenen Variablen vor (vgl. Abb. 15.10a). Bei der Extraktion von Daten aus einer entsprechenden Datenbank entstehen Ergebnistabellen, die für jede darin enthaltene Kombination aus Station und Variable jeweils eine eigene Zeile enthalten. Dokumentiert das Ergebnis mehrere Variablen für einzelne Stationen, ist in diesem beispielsweise der Name dieser Stationen jeweils mehrfach enthalten (vgl. Abb. 15.10b). Da die Versendung solcher redundanter Informationen das Datenvolumen erhöht und somit zu Lasten der Übertragungsgeschwindigkeit geht, kann die interne Ergebnisrepräsentation so konfiguriert werden, dass sie die in ihm enthaltenen Daten in 1:n-Relationen[264] repräsentiert (vgl. Abb. 15.10c). Auf diese Weise kann eine weitgehend redundanzfreie Ergebnisübertragung zum jeweiligen Clientrechner erfolgen. Da eine Limitierung der pro Anfrage zu versendenden Datensätze erfolgt (vgl. Kap. 15.2.1), wird dabei sicherge-

[264] Die von der interne Ergebnisrepräsentation bereitgestellte Sortierfunktionalität unterstützt in diesem Fall auch das Sortieren der *variablenspezifischen* Ausprägungen individueller Stationen, so dass diese für jede Station bspw. in eine alphabetische Reihenfolge gebracht oder anhand des Verfügbarkeitszeitraumes sortiert werden können.

stellt, dass sämtliche zu extrahierenden Datensätze zu einer konkreten Station genau einem Anfrageergebnis zugeordnet werden. Auf diese Weise wird verhindert, dass sich die gemäß der vorliegenden Selektionskriterien bereitzustellenden Datensätze für eine Station gegebenenfalls auf zwei Teilergebnisse aufteilen – und damit relevante Informationen möglicherweise übersehen werden.

15.3 Entwurf der Kernkomponente *Auswertung*

15.3.1 Kommunikation zwischen unabhängigen Auswertungsmodulen

Für eine effiziente Auswertung von Ergebnisdaten ist es wünschenswert, dass bestimmte Interaktionen - bspw. zur Auswahl einer Ergebnisuntermenge - , die der Anwender über ein Auswertungsmodul ausführt, nicht nur dort zu einer veränderten Darstellung der Daten führen, sondern zugleich die Darstellung in weiteren aktiven Auswertungsmodulen verändern[265]. Eine hierfür an sich sinnvolle Kopplung einzelner Module limitiert allerdings ihre erforderliche wechselseitige Unabhängigkeit und schränkt somit ihre flexible Verwendbarkeit ein. Um diese Limitierungen zu umgehen, wird eine Kommunikationsstruktur verwendet, die sich an objektorientierten Entwurfsmustern orientiert, die eine Wiederverwendbarkeit graphischer Oberflächen unterstützen. Hierzu zählen das Model-View-Controller (MVC)-Muster (vgl. bspw. [Balzert 1999, 692ff.]) mit einer klaren Trennung zwischen Datenrepräsentation (Model), eigentlicher Nutzerschnittstelle (View) und Funktionalität (Controller) sowie das Observer-Pattern (vgl. bspw. [Balzert 1999, 694ff.]), das als Vereinfachung des MVC-Musters durch Zusammenfassung von View und Controller angesehen werden kann. Die Unabhängigkeit der zusammenwirkenden Komponenten voneinander wird durch geeignete Kommunikationsschnittstellen erreicht, die es einer Komponente bspw. erlauben, andere Komponenten über Zustandsänderungen zu unterrichten.

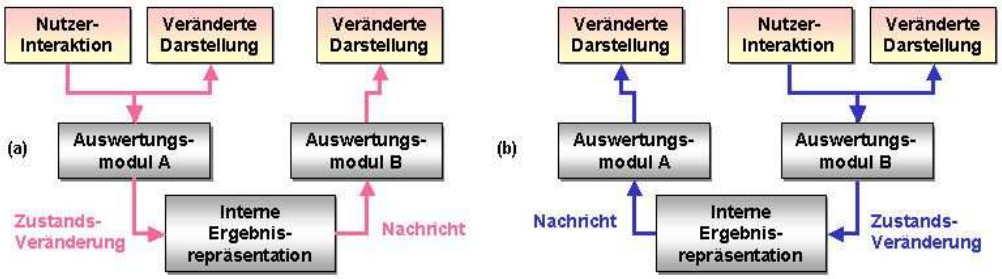

Abb. 15.11 - Interaktion und Benachrichtigung zwischen unterschiedlichen Auswertungsmodulen und der internen Ergebnisrepräsentation: Eine Nutzerinteraktion über Auswertungsmodul A verändert den Zustand in der internen Ergebnisrepräsentation und zugleich die Darstellung in beiden Auswertungsmodulen (a) und *vice versa* (b).

Der hier gewählte Ansatz kann vereinfacht als eine autonome Interaktion jedes aktiven Auswertungsmoduls mit einer internen Ergebnisrepräsentation in Kombination mit einer automatischen Benachrichtigung aller anderen aktiven Auswertungsmodule über dadurch auftretende Zustandsänderungen beschrieben werden. Sind etwa zwei Auswertungsmodule für gleichzeitige, jeweils unterschiedliche Darstellungen eines Anfrageergebnisses aktiv und wird über eines dieser Module beispielsweise die Selektion eines Datensatzes aus dem Ergebnis durchgeführt, führt dies zu einer Zustandsveränderung in der internen Ergebnisrepräsentation, die in der Folge auch im jeweils anderen Auswertungsmodul sichtbar wird. Abb. 15.11 verdeutlicht dieses Prinzip.

[265] Ein typisches Beispiel ist die Selektion von Datensätzen in einer textuellen Darstellung und die automatische Hervorhebung dieser Datensätze in einer Visualisierung und umgekehrt (vgl. Kap. 21.2).

15.3.2 Architektur

Die Architektur der Kernkomponente *Auswertung* (vgl. Abb. 15.12) setzt sich aus drei zentralen Bestandteilen zusammen – einem SetupManager zu ihrer Steuerung gemäss ihrer Konfiguration für einen individuellen Datenraum, einem ResultManager zur internen Bereitstellung der aktuellen Ergebnisdaten sowie aus dem Satz der jeweils aktiven Auswertungsmodule zur Interaktion des Anwenders mit den Daten.

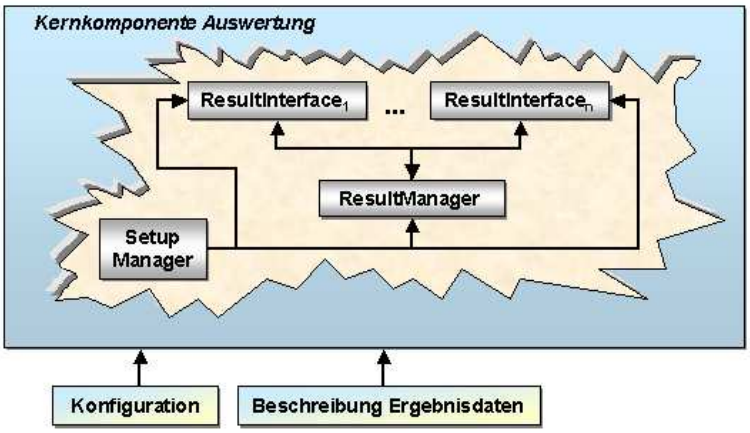

Abb. 15.12 - Architektur der Kernkomponente *Auswertung*.

- **SetupManager**

Der SetupManager der Kernkomponente *Auswertung* dient zur Umsetzung ihrer Konfiguration auf den jeweiligen Datenraum. Er steuert die Zusammensetzung und Adaption der zu verwendenden Auswertungsmodule sowie das Verhalten des für die interne Verwaltung der Ergebnisdaten zuständigen ResultManager.

- **ResultManager**

Die Aufgabe des ResultManager ist die zentralen Verwaltung und Bereitstellung der jeweils aktuell vorliegenden Ergebnisdaten. Er stellt sie den aktiven Auswertungsmodulen zur Verfügung und steuert die Kommunikation zwischen diesen.

- **ResultInterfaces**

Die einzelnen Auswertungsmodule, die für einen Datenraum aktiviert werden, werden intern als ResultInterfaces bezeichnet. Jedes ResultInterface stellt dem Anwender spezifische Formen der Darstellung und Interaktion mit den Ergebnisdaten zur Verfügung, auf die es jeweils über den ResultManager zugreift.